Deep reinforcement learning for supply chain synchronization

Ilya Jackson
Massachusetts Institute of Technology, Center for Transportation and Logistics, Cambridge, United States
ilyajack@mit.edu

Abstract

Supply chain synchronization can prevent the "bullwhip effect" and significantly mitigate ripple effects caused by operational failures. This paper demonstrates how deep reinforcement learning agents based on the proximal policy optimization algorithm can synchronize inbound and outbound flows if end-to-end visibility is provided. The paper concludes that the proposed solution has the potential to perform adaptive control in complex supply chains. Furthermore, the proposed approach is general, task unspecific, and adaptive in the sense that prior knowledge about the system is not required.

1. Introduction

The 21st century has begun in a fruitful environment of growing computational power that is becoming cheaper and widely available over time. This tendency, along with the increasing quantities of easily-to-harvested data, creates perfect conditions for algorithms capable of leveraging the increased computational resources [1]. In recent years, deep learning has been the most highly debated topic in artificial intelligence. It has been applied in a broad range of fields vital for supply chain management and logistics, including image recognition and natural language processing. This success was mainly achieved within the supervised learning paradigm [2]. However, the potential of deep learning extends classical supervised learning. Deep reinforcement learning (DRL) is the paradigm of choice for applying deep learning to non-supervised scenarios. A neural network trained by reinforcement learning can be given an input that has not been encountered before. It then returns the valid output, which means that the underlying structure of the input has been learned. This is analogous to how a child learns to interact with the physical world. By exploring possible sequences of actions and associating them with long-term rewards, an adaptive controller based on this principle can learn how to operate in dynamic and non-deterministic environments. Such an adaptive controller is known as a DRL agent.

DRL has demonstrated notable performance in the fields of autonomous aerial vehicles [3], road traffic navigation [4], autonomous vehicles [5], and robotics [6]. Besides, in recent studies, DRL showed solid capabilities of mastering video games that partially capture and reflect the complexity of the real world. Complexity in this context includes long-term planning horizons, partial observability, imperfect information, and high dimensionality of state and action spaces. Examples of such games include Atari games [7], Go [8], Dota 2 [9], StarCraft 2 [10], and Minecraft [11].

It is awe-inspiring that AlphaGo, a DRL-based program developed by DeepMind, defeated one of the most celebrated Go players in the world, Lee Sedol, in four out of five games. Skeptics can undervalue the astonishing success of AlphaGo and AlphaZero in mastering such classic board games as chess, shogi, and Go through self-play by pointing out the relative simplicity of rules and determinism of games. However, it is worth highlighting that the algorithm was able to learn the game of Go from scratch, starting with random play. It then played against itself multiple times and each time improved its play by using DRL to select moves that led to more successful outcomes [8]. However, could DRL demonstrate the same success in real-world domains like supply chain management?

In the past, supply chain management was mainly focused on the management of inventory, transportation, and operations. However, in recent years, the focus has shifted to the management of information and data flows. This is because the supply chain is now more than just a physical system of transportation and inventory. It is also an information system that requires the coordination of many different parties. Nowadays, supply chains are highly complex systems of interconnected processes that require prompt business-critical decisions to stay competitive and adaptive in dynamic environments. Adaptive control in

URI: https://hdl.handle.net/10125/79578
978-0-9981331-5-7

such systems ensures delivery to end customers with minimal delays and avoids extra costs. In order to achieve this objective, production scheduling, inventory control, transportation plans, and last-mile delivery must be synchronized among a multitude of individual supply chain participants that frequently spread across the globe.

Since inventory level is the difference between inbound and outbound material flows, inventory control is a pivotal element of supply chain synchronization [12]. The disruptive rise-and-fall inventory dynamic widely known as the "bullwhip effect" [13] and ripple effects caused by operational failures [14] are, on the other hand, notable consequences of supply chain desynchronization. The bullwhip effect is a significant problem for supply chain managers because it eventually causes over- and under-production cycles, which leads to excess inventory or stockouts.

In order to understand the mechanics behind this phenomenon, let's assume that usually, a retailer sells 100 commodity units each day and has an inventory level sufficient for four days, namely 400 units. However, suppose the retailer experiences a surge in selling 200 units. In that case, the retailer will order at least 200 units just to replace what is sold, but it is absolutely natural for the retailer to expect the sales surge to last for a while or even shape a new upward trend. In this case, it may order another 400 units (600 in total) to make sure that it will have four days' worth of inventory at the new units-per-day sales rate. The distributor then observes a jump in orders and places substantially higher orders for the factory, so initial demand fluctuation propagates and amplifies upstream in the supply [15].

A DRL agent has the potential to perform adaptive coordination along the whole supply chain if information transparency and end-to-end visibility is ensured. Pandemic shocks and disruptions at both strategic and managerial levels, along with post-pandemic recoveries, can become a catalyst for necessary changes in data transparency and global supply chain coordination [16].

Given these facts, the following research question arises: "How can a reinforcement learning agent synchronize inbound and outbound flows in a stochastic supply chain environment assuming that end-to-end visibility is provided?"

2. Related work

This section sheds light on the related work in bullwhip effect studies and DRL applications for supply chain coordination.

2.1. Bullwhip effect in supply chains

The bullwhip effect, also widely known as the Forrester effect and information amplification, has been a subject of research for almost 60 years [17]. Therefore, the complete literature overview is beyond the scope of this paper. Nevertheless, it is worth highlighting several studies.

Traditionally the bullwhip effect studies have mainly focused on the operational level. For example, Lin et al. studied the causes and mitigation solutions, including reducing lead time and increasing information transparency [18]. In addition, Wang and Disney developed a simulation model to consider various assumptions and approximations for modeling the bullwhip effect concerning demand, forecast, delay, and replenishment [19]. However, a multitude of studies addressed the behavioral causes behind the bullwhip effect. These studies include topics related to inventory information sharing [20], trust in partnership [21], and the human factor in forecasting [22]. These works incorporated behavioral aspects and shaped an alternative research approach to the bullwhip effect [23].

In contrast to these studies, we discuss how adaptive coordination along the supply chain can be performed algorithmically if end-to-end visibility is ensured.

2.2. Deep reinforcement learning in supply chain management and inventory control

Among DRL applications to supply chain management and multi-echelon inventory control, it is worth highlighting [24], which demonstrated the efficiency of Q-learning for dynamic inventory control in a multi-echelon supply chain model. Barat et al. presented a DRL agent based on the actor-critic architecture for closed-loop replenishment control in supply chains [25]. Zhao et al. adapted the Soar RL algorithm and modeled the problem as an asymmetrical wargame environment to reduce the operational risk across the supply chain [26]. Wang et al. applied a DRL agent to the supply chain coordination problem under uncertainty [27]. Chen et al. proposed the DRL framework for effective management for blockchain-based [28]. Perez et al. compared several reinforcement learning and heuristic methods, including DRL, on a single product inventory-control problem under stochastic stationary consumer demand [12].

A DRL agent capable of playing the beer distribution game was proposed in the recent study [29]. The beer distribution game is commonly used in supply chain management education to demonstrate the importance of supply chain coordination. The DRL

agent is based on deep Q-learning and has been trained without the provision of any preliminary data on costs structure or other settings of the simulated environments.

In the numerical experiments following in the next chapters, the environment is built upon OR-Gym [30], an open-source library containing operations research problems in the form of RL environments. A DRL agent uses the Proximal Policy Optimization (PPO) algorithm. PPO does not require exhaustive hyperparameter tuning and thus promises to be a general, task unspecific, and capable of performing adaptive control in complex supply chains. The recent paper provides a detailed technical description of the implementation and configurations [31].

3. Methodology

This section presents the core methodology behind RL and Markov Decision Process (MDP). Shortly after that, the supply chain environment (SCE) is introduced. Lastly, the section describes the PPO algorithm.

3.1. Reinforcement learning

Supervised learning assumes learning from data of labeled observations. Therefore, the key objective behind supervised learning is to generalize and respond to previously unobserved information. In contrast, RL in general and DRL in particular, are goal-directed approaches. DRL agent explores the surrounding environment by trial and error in order to find the optimal policy. Optimality in this context is measured by a cumulative reward, a numerical measure that describes the goodness of action in a particular state. (Figure 1). SCE can be naturally formulated as MDP. MDP acts as the flexible mathematical framework for goal-directed learning and can be described as a tuple $(S, A, p(.), R, \gamma)$.

The DRL agent and environment interact at each instance of a sequence of discrete-time steps $t \in T$. At time step t, the agent receives a representation of the environmental state $S_t \in S$ and reward $R_t \in R$ in response to performed action $A_t \in A$. After that, the agent moves to a new state S_{t+1}. States, actions, and rewards produce a sequence known as trajectory. The sequence follows the following pattern $S_0, a_0, R_1, S_1, A_1, \ldots, S_n$, where S_n stands for the terminal state. At each time step, the agent performs multiple trials with the stochastic environment (episodes).

The function $p(.)$ defines the dynamics of the MDP and can be described by the following equation:

$$p(S, A^*) = Pr\{S_{t+1} = S^*,\ R_{t+1} = R^* | S_t = S,\ A_t = A^*\} \quad (1)$$

The agent observes S_t and R_t performing actions under a policy π. In such settings, the agent's goal is to find the optimal policy that maximizes the expected return from each state S_{t+1}. The expected return at time t is defined as $R_t = E[\sum_{k=0} \gamma^k r_{t+k}]$, where $\gamma \in [0, 1]$is the discount factor that determines the tradeoff between immediate and distant rewards.

In SCE, the agent must make a decision regarding the order size at each stage m and each time step t. The action a_t is an integer value that corresponds to order size at each supply chain stage. The state S_t is a vector that includes the current inventory levels for each stage as well as the history of previously taken actions. Thus, the RL agent attempts to synchronize demand and supply in order to maximize the total revenue over the time horizon (Equation 7).

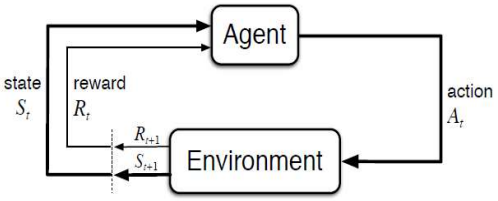

Figure 1. Agent-environment relations in a Markov decision process [32].

3.2. Supply chain environment

Supply chains are incredibly complex. Each step of the supply chain is crucial from both operational and strategic points of view, and each step can impact other components of the supply chain. Supply chain disruptions and failures can cause significant problems for a company and its business partners. However, in the simplified form, a supply chain can be considered a system for delivering a product from its origin to the end-user. SCE is based on the seminal work with such a simplification [33].

SCE simulates a single-product multi-echelon supply chain model. The model makes several assumptions. First, the tradable goods do not perish over time, and replenishment sizes are integers. Second, intermediaries involved in the supply chains can be represented as stages $M = \{0, 1, \ldots m_{end}\}$. Stage 0 stand for a retailer that fulfills arising customer's demand. Stage m_{end} represents a raw material supplier. Stages from 1 to m_{end} - 1 represent mediators involved in the product lifecycle, for example, distributors and producers (Figure 2). One commodity unit from the previous stage is transformed into one unit at the following stage until the final product is manufactured and transported. Replenishment lead times between

stages are constant, integer, and measured in days. Besides, lead times take into consideration both production and transportation. Production capacities and storage levels are limited for all the stages, except the last one. During the simulation run at each time step $t \in T$, the following sequence of events occurs:

1. All the stages except the raw material pool place orders.
2. Replenishment orders are fulfilled according to available inventory capacity.
3. Demand is satisfied according to the inventory availability at stage 0 (retailer).
4. Backlogging is not allowed.
5. Holding costs are charged for each unit of inventory on hand.

$\forall\, m \in M$ and $\forall\, t \in T$, the SCE dynamics is governed by the following set of equations:

$$I_{t+1}^{m} = I_{t}^{m} + Q_{t-Lm}^{m} - \zeta_{t}^{m} \quad (2)$$

$$V_{t+1}^{m} = V_{t}^{m} - Q_{t-Lm}^{m} + Q_{t}^{m} \quad (3)$$

$$Q_{t}^{m} = min(c^{m+1}, I_{t}^{m+1}, \hat{Q}_{t}^{m}) \quad (4)$$

$$\zeta_{t}^{m} = \begin{cases} Q_{t}^{m-1}, if\ m > 0 \\ \min(I_{t}^{0} + Q_{t-Lm}^{0}, D_{t}),\ if\ m = 0 \end{cases} \quad (5)$$

$$U_{t}^{m} = \hat{Q}_{t}^{m-1} - \zeta_{t}^{m} \quad (6)$$

$$Net_{t}^{m} = \rho^{m}\zeta_{t}^{m} - r^{m}Q_{t}^{m} - k^{m}U_{t}^{m} - h^{m}I_{t+1}^{m}, \quad (7)$$

where at the beginning of each period $t \in T$ at each stage $m \in M$, I denotes inventory on hand. V stands for the pipeline inventory (commodities on the way). Q corresponds to the accepted reorder quantity and $\hat{Q}$ is the reorder quantity. L is replenishment lead time between stages. D stands for demand, a discrete random variable under Poisson distribution [34]. Sales ζ at each period equal to the satisfied customer demand at stage 0. U corresponds to unfulfilled demand and unfulfilled reorder requests. The net profit *Net* equals sales profit minus procurement costs, the penalty for unsatisfied demand, and storage costs. ρ, r, k, and h stand for the unit sales price, unit procurement cost, unit penalty for unsatisfied demand, and unit holding costs. If production capacity and inventory level do not exceed capacity constraints c, Q would be equal to $\hat{Q}$. However, if capacities are insufficient, it sets an upper bound on the order size that can be accepted. Additionally, it is assumed that the capacities at stage m_{end} are infinite.

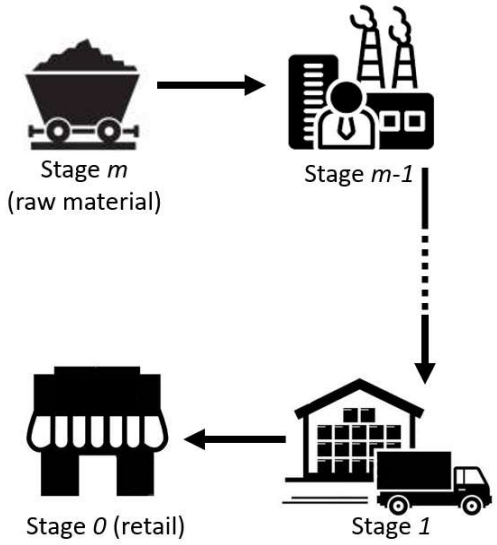

Figure 2. Supply chain with m stages.

3.3. Proximal Policy Optimization algorithm

A DRL agent uses PPO, an algorithm developed by the OpenAI team and distinguished by simple implementation, generality, and low sample complexity. The sample complexity denotes the number of training samples that the RL agent requires to learn an objective function. In 2017, OpenAI used PPO to create a version of the Dota 2 bot, which beat the world champion, thereby demonstrating that a single system can learn from self-play to match or exceed the skill of the best human professionals at one of the most popular video games [9].

It is imperative to highlight that the vast majority of machine learning algorithms in general and DRL algorithms, in particular, require hyperparameters that must be finetuned. Thus, an extra advantage of PPO is its capability of providing decent results with default configuration or relatively simple hyperparameter tuning [35].

PPO belongs to a broad class of actor-critic approaches. It uses two deep artificial neural networks, one to suggest actions at each time step (*actor*) and one to predict the corresponding rewards (*critic*). The agent interacts with an environment through continuous actions. At each time step, the agent uses one of its available actions to generate an observation in the environment, which it then receives in the form of a reward. The goal is to perform actions that maximize the expected long-term cumulative reward. The actor learns a policy that produces a probability distribution of available actions. The critic, in its turn, learns to estimate the reward function for given state-action pairs. The loss function *Loss(θ)* reflects the difference between the reward predicted by the critic with

parameters θ and the actual reward received from the environment (SCE in the following numerical examples). PPO limits the update of the parameters by clipping the loss function (Equation 8). This loss function is famous in the DRL community for stable learning compared to other state-of-the-art policy gradient methods across various benchmarks.

$$Loss(\theta) = min[\frac{\pi_{k-1}}{\pi_k}\hat{A}_t, cl(\frac{\pi_{k-1}}{\pi_k}, 1 - \varepsilon, 1 + \varepsilon)\hat{A}_t], (8)$$

where π_{k-1} and π_k stand for the previous policy and the new policy. k is the number of updates to the policy since initialization. The function *cl(.)* imposes the constraint of the form $1 - \varepsilon \leq \pi_{k-1} / \pi_k \leq 1 + \varepsilon$, where ε is a tunable hyperparameter that prevents extreme policy updates. The advantage estimation of the state is the sum of the discounted prediction errors over T time steps $\hat{A}_t = \sum \gamma^{T-t+1}\delta_T$, where δ_T is the difference between the actual and the estimated rewards, also known as the temporal difference error. γ stands for the discount rate.

4. Experiment

This section describes two numerical experiments with different environment configurations. The simulation runs last for 30 modeling days. Furthermore, in both experiments, a DRL agent based on PPO is compared with the base-stock policy, a common approach in classic inventory control theory. The number of experiments is limited to two due to the computational burden of the current implementation. Taking into account an ongoing reproducibility crisis in science [36], all the experiments are conducted in a reproducible manner using Google Colaboratory, a hosted version of Jupyter Notebooks [37]. Therefore, results can be reproduced and verified by anyone. Such a cloud-based implementation imposes hardware limits, namely GPU memory of 12 GB, GPU memory clock of 0.82 GHz, and GPU performance of 4.1 teraflops. Nevertheless, these experiments can be considered sufficient at the proof-of-concept stage.

4.1. Implementation and configurations

The first numeric experiment is conducted with a 4-stage supply chain. Table 1 contains the parameters of SCE used in the experiment. The second numeric experiment is performed with a 5-stage supply chain using more challenging environment settings (Table 2). The PPO algorithm is implemented in parallel using the Ray framework. The framework performs actor-based computations governed by a dynamic execution engine. Besides, Ray is equipped with a distributed scheduler and fault-tolerant storage framework [38]. In the numerical experiments, a relatively simple feed-forward architecture with one hidden layer and 256 neurons is used for both actor and critic artificial neural networks. An exponential linear unit is u as an activation function, ε equals to 0.32, and the learning rate is 10^{-5}.

Table 1. Parameters values for the first numerical experiment.

Parameter	Stages (m)			
	0	1	2	3
D	20	-	-	-
I_0	100	100	200	∞
P	2	1.5	1	0.5
R	1.5	1.0	0.75	0.5
k	0.1	0.075	0.05	0.025
h	0.15	0.10	0.5	-
c	-	100	90	80
L	3	5	10	-

Table 2. Parameters values for the second numeric experiment.

Parameter	Stages (m)				
	0	1	2	3	4
D	20	-	-	-	-
I_0	100	120	150	150	∞
ρ	4.1	2	1.5	1	0.5
r	1.0	1.0	0.75	0.5	0.5
k	0.10	0.075	0.05	0.025	0.025
h	0.10	0.10	0.05	0.05	-
c	-	120	100	100	90
L	3	5	7	7	-

The PPO algorithm is compared with the base-stock policy, a classic operations research approach to inventory control [39]. This comparison can be considered sufficient at the proof-of-concept stage. However, in future research, it is worth comparing DRL with a broader spectrum of techniques.

The numeric experiment is fully reproducible within Google Colaboratory. Thus, results can be reproduced and verified by anyone [37].

4.2. Results

In the first experiment, the base-stock policy used as a benchmark could produce a policy with an average reward of 414.3 monetary units. On the other hand, in 66000 training episodes, the PPO agent could derive a policy that leads to a mean reward of 425.6 monetary units, which is 2.7% higher. The learning path and comparison against the base-stock benchmark are demonstrated in Figure 3. Figure 4 illustrates the

inventory dynamics for two weeks at all the stages under the control of two algorithms.

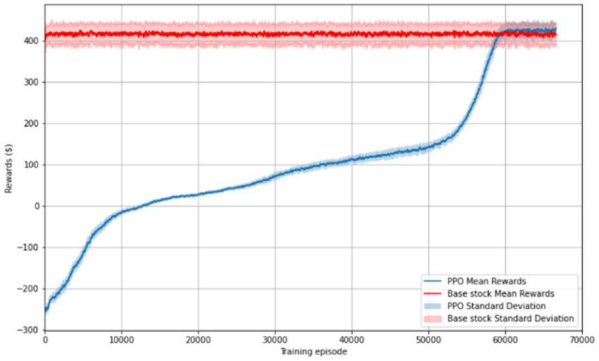

Figure 3. Learning curve. The performance of the PPO agent increases during training episodes.

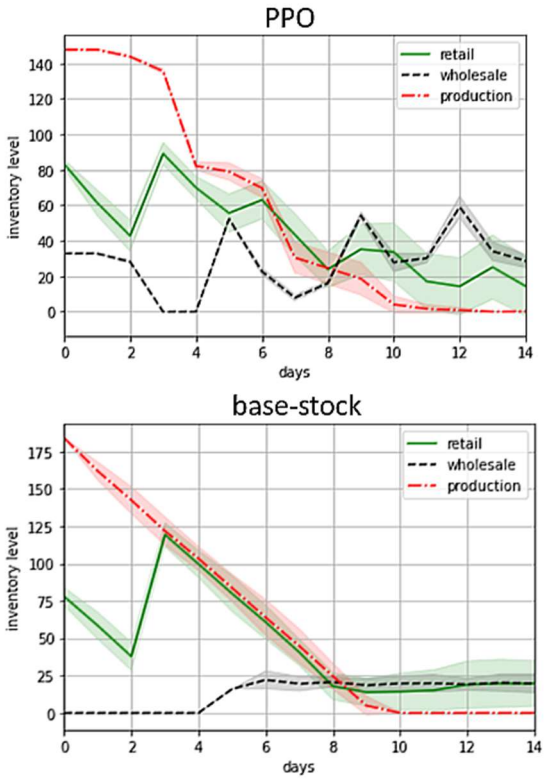

Figure 4. Inventory dynamics produced by the trained PPO agent and the base-stock policy.

During the second experiment, unusual behavior was observed. The PPO agent uses the policy that terminates placing new orders closer to the end of the simulation run. This phenomenon can be explained by recalling the fact that the simulation time is limited by 30 days, and the agent predicts that there will not be enough time to sell all the goods. Holding costs associated with extra inventory levels will not eventually pay off. Figure 5 illustrates this unusual behavior under the control of the trained PPO agent. Perhaps, such a problem may be avoided or at least mitigated by modifying the SCE, for example, by adding a "cool down" period. However, this modification would require significant changes in experiment design and algorithmic implementation and, therefore, deferred for future research.

The tendency of the DRL agent to exploit the underlying mechanics of simulated environments is common and well-known in the domains of cybersport and virtual reality. Therefore, this discovery deserves specific attention during the development of applications for real-world problems.

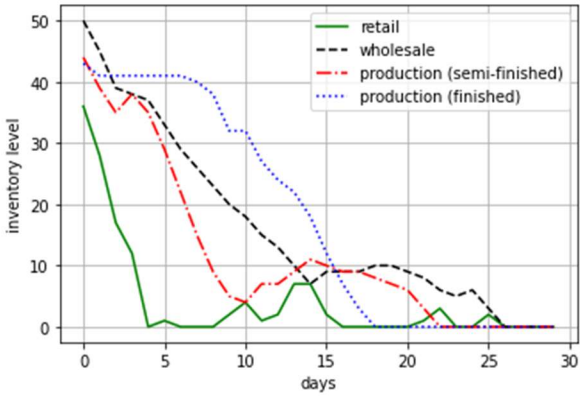

Figure 5. Unexpected inventory dynamics caused by the PPO agent.

Nevertheless, the PPO agent managed to adapt and find a profitable policy in the specified environmental setup, even with an unpredictable solution. On the other hand, the policy under base-stock did not entail a profitable outcome (Figure 6).

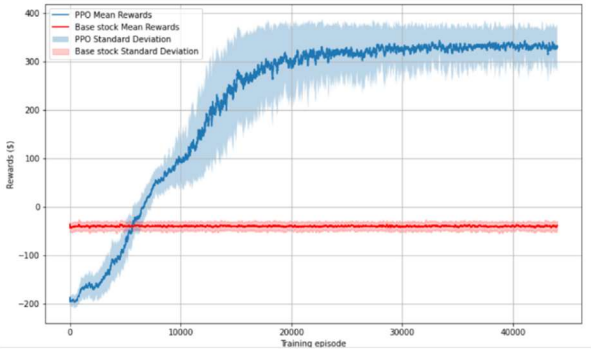

Figure 6. The PPO agent decisively outperforms the base-stock policy under the more challenging environment settings.

Table 3 summarizes the results of both experiments. The table contains mean rewards, standard deviations of

rewards, and 95% confidence intervals (CI). The values are reported for the thoroughly trained DRL agent and base-stock policy.

Table 3. Summary of experiments.

Experiment №	Mean	Std.	CI (95%)
1 (PPO)	425.6	19.4	425.6 ± 3.12
1(base-stock)	414.3	26.5	414.3 ± 4.75
2 (PPO)	328.7	93.1	328.7 ± 16.5
2(base-stock)	−43.4	7.6	−43.4 ± 1.4

5. Discussion

The proposed solution has demonstrated convincing capabilities of performing adaptive control in multi-stage supply chains. The solution relies on a DRL agent that synchronizes the supply chain by interacting with a simulated environment. Taking advantage of the PPO algorithm, the proposed solution appears to be general, universal, task-unspecific and does not rely on prior knowledge on the virtual environment. Besides, it is straightforward to implement and train. However, the reliance on the simulated environments requires specific attention. DRL agents can coordinate the supply chain if and only if end-to-end visibility is provided. Additionally, planning and real-time control require data availability on replenishment, inventory levels, demand, and production capacities [40]. The ultimate satisfaction of these requirements is an advent of a digital replica of a supply chain capable of representing the system state for any given moment in real-time. This concept has been recently introduced and is currently known as a digital supply chain twin [16]. Incorporating DRL agents into supply chain digital twins can be considered one of the most promising directions for future research.

The SCE should not be oversimplified and must capture all the critical relations within real-world supply chain elements. The tendency of the DRL agent to exploit simulation mechanics has been discovered in this study and deserves primary consideration in development and validation. Therefore, it is crucial to highlight that many advanced supply chain models exist as discrete-event simulation models [41]. Nowadays, the vast majority of DRL research follows the OpenAI Gym standard [42]. In short, OpenAI gym is a toolkit for developing and comparing reinforcement learning algorithms. An RL environment has to be a Python class inheriting from *gym.Env* and containing the *step* method. The *step* method takes action as the input argument and returns the next state and the corresponding reward value. Thus, unlike in discrete-event simulation models, the time by default advances with a constant time step. Therefore, in order to transform discrete-event simulation into a standardized DRL environment, the time has to become a state variable directly observable by the agent. This problem was mentioned by Schuderer et al. [43] and, if a standardized and straightforward way of transforming discrete-event simulation models into DRL environments is developed, a manifold of the models developed over the past 30 years will become immediately available to RL applications.

5. Conclusions

The PPO algorithms can perform adaptive control in complex supply chains and have the potential to solve supply chain synchronization problems. The presented numerical experiments showed the capability to outcompete the base-stock policy. However, they also revealed the tendency of the learning agent to exploit simulation mechanics. Among the technical advantages, it is critical to highlight that DRL agents based on PPO are general and simple to finetune. Additionally, the implementation and training are straightforward using contemporary frameworks.

From the industrial applicability point of view, the assumption of the end-to-end visibility is unrealistic in many real-world settings given the current digitalization level, and the adoption of the fully-fledged supply chain digital twins can be considered a mandatory infrastructural condition.

6. References

[1] R. Hamerly, "The Future of Deep Learning Is Photonic: Reducing the energy needs of neural networks might require computing with light". IEEE Spectrum 58(7), 2021, pp. 30-47.

[2] Goodfellow, I., Y. Bengio, A. Courville, and Y. Bengio, Deep learning. 2nd edn. Cam-bridge, MIT press, 2016.

[3] A. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, E. Liang, "Autonomous inverted helicopter flight via reinforcement learning", Experimental Robotics 9, 2006, pp. 363-372.

[4] L. Fridman, J. Terwilliger, B. Jenik, "Deeptraffic: Crowdsourced hyperparameter tuning of deep reinforcement learning systems for multi-agent dense traffic navigation". arXiv preprint arXiv:1801.02805, 2018.

[5] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning". In: Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 2034-2039.

[6] S. Gu, E. Holly, T. Lillicrap, S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates". In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2017), 2017, pp. 3389-3396.

[7] K. Clary, E. Tosch, J. Foley, and D. Jensen, “Let's Play Again: Variability of Deep Reinforcement Learning Agents in Atari Environments”. arXiv preprint arXiv:1904.06312, 2019.

[8] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”. Science 362, 2018, pp. 1140-1144.

[9] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Jozefowicz, “Dota 2 with large scale deep reinforcement learning”. arXiv preprint arXiv:1912.06680, 2019.

[10] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, “Grandmaster level in StarCraft II using multi-agent reinforcement learning”. Nature 575, 2019, pp. 350-354.

[11] W.H. Guss, C. Codel, K. Hofmann, B. Houghton, N. Kuno, S. Milani, S.P. Mohanty, D.P. Liebana, R. Salakhutdinov, N. Topin, M. Veloso, P. Wang, “The MineRL Competition on Sample Efficient Reinforcement Learning using Human Priors”. arXiv preprint arXiv: 1904.10079, 2019.

[12] H.D. Perez, C.D. Hubbs, C. Li, I.E. Grossmann, “Algorithmic Approaches to Inven-tory Management Optimization”. Processes 9 (1), 2021, p. 102.

[13] H.L. Lee, V. Padmanabhan, S. Whang, “Information distortion in a supply chain: The bullwhip effect”. Management science 43(4), 1997, pp. 546-558.

[14] Y. Li, K. Chen, S. Collignon, D. Ivanov, “Ripple effect in the supply chain network: Forward and backward disruption propagation, network health and firm vulnerability”. European Journal of Operational Research 291(3), 2021, pp. 1117-1131.

[15] Y. Sheffi, The New (Ab) Normal: Reshaping Business and Supply Chain Strategy Beyond Covid-19. MIT CTL Media, 2020.

[16] D. Ivanov, A. Dolgui, “A digital supply chain twin for managing the disruption risks and resilience in the era of Industry 4.0”. Production Planning & Control, 2020, 1-14.

[17] J. Forrester, “Industrial dynamics”. Journal of the Operational Research Society 48(10), 1997, pp. 1037-1041.

[18] J. Lin, M. Naim, L. Purvis, J. Gosling, “The extension and exploitation of the inventory and order based production control system archetype from 1982 to 2015”. Int. J. Prod. Econ. 194, 2017, pp. 135-152.

[19] X. Wang, S. Disney, “The bullwhip effect: progress, trends and directions”. Eur. J. Oper. Res. 250 (3), 2016, pp. 691-701.

[20] R. Croson, K. Donohue, E. Katok, J. Sterman, “Order stability in supply chains: coordination risk and the role of coordination stock”. Prod. Oper. Manag. 23 (2), 2014, pp. 176-196.

[21] Q. Cao, J. Baker, D. Schniederjans, “Bullwhip effect reduction and improved business performance through guanxi: an empirical study”. Int. J. Prod. Econ. 158, 2014, pp. 217-230.

[22] P. Baecke, S. De Baets, K. Vanderheyden, “Investigating the added value of integrating human judgment into statistical demand forecasting systems”. Int. J. Prod. Econ. 191, 2017, pp. 85-96.

[23] Y. Yang, J. Lin, G. Liu, and L. Zhou, “The behavioural causes of bullwhip effect in supply chains: A systematic literature review”. International Journal of Production Economics, 2021, pp.108–120.

[24] A. Mortazavi, A. Arshadi Khamseh, P. Azimi, “Designing of an intelligent self-adaptive model for supply chain ordering management system”. Engineering Applications of Artificial Intelligence 37, 2015, pp. 207-220.

[25] S. Barat, H. Khadilkar, H. Meisheri, V. Kulkarni, V. Baniwal, P. Kumar, M. Gajrani, “Actor based simulation for closed loop control of supply chain using reinforcement learning”. In: Proceedings of the 18th International Conference on Autonomous Agents and Multi Agent Systems, 2019, pp. 1802-1804.

[26] Y. Zhao, E. Hemberg, N. Derbinsky, G. Mata, and U.M. O’Reilly, “Simulating a Logistics Enterprise Using an Asymmetrical Wargame Simulation with Soar Reinforcement Learning and Coevolutionary Algorithms”, 2021.

[27] R. Wang, X. Gan, Q. Li, and X. Yan, “Solving a Joint Pricing and Inventory Control Problem for Perishables via Deep Reinforcement Learning”. Complexity, 2021.

[28] H. Chen, Z. Chen, F. Lin, and P. Zhuang, “Effective Management for Blockchain-Based Agri-Food Supply Chains Using Deep Reinforcement Learning”. IEEE Access 9, 2021, pp. 3608-3618

[29] A. Oroojlooyjadid, M. Nazari, L.V. Snyder and M. Takac, “A Deep Q-Network for the Beer Game: Deep Reinforcement Learning for Inventory Optimization”. Manufactur-ing & Service Operations Management, 2021, pp. 1-20.

[30] C. Hubbs, H.D. Perez, O. Sarwar, N.V. Sahinidis, I.E. Grossmann, J.M. Wassick, “OR-Gym: A Reinforcement Learning Library for Operations Research Problem”. arXiv preprint arXiv:2008.06319, 2020.

[31] Z. Kegenbekov, and I. Jackson, “Adaptive Supply Chain: Demand–Supply Synchronization Using Deep Reinforcement Learning. Algorithms” 14(8), 2021, p.240.

[32] Sutton, R.S., A.G. Barto, Reinforcement learning: An introduction. MIT press, 2018.

[33] P. Glasserman, S. Tayur, “Sensitivity analysis for base-stock levels in multiechelon production-inventory systems”. Management Science 41(2), 1995, pp. 263–281.

[34] R. Teunter, W. Haneveld, “Dynamic inventory rationing strategies for inventory systems with two demand classes, Poisson demand and backordering”. European journal of operational research 190(1), 2008, pp. 156-178.

[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, “Proximal policy opti-mization algorithms”. arXiv preprint arXiv:1707.06347, 2017.

[36] M. Baker, Reproducibility crisis. Nature 533(26), 2016, pp. 353-66.

[37] Executable Colab Notebook: Google Colaboratory, https://colab.research.google.com/drive/1D_E1j10skboh

OOA4vbuy9OCw4dqbAW_S?usp=sharing, last accessed 2021/9/19.
[38] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M.I. Jordan, I. Stoica, "Ray: A distributed framework for emerging {AI} applications". In: Proceedings of the 13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18), 2018, pp. 561-577.
[39] R. Kapuscinski and S. Tayur, "Optimal policies and simulation-based optimization for capacitated production inventory systems". In Quantitative Models for Supply Chain Management, Springer, 1999, pp. 7-40.
[40] F. Saifutdinov, I. Jackson, J. Tolujevs, and T. Zmanovska, Digital Twin as a Decision Support Tool for Airport Traffic Control. In 2020 61st International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS), IEEE, pp. 1-5, 2020.
[41] I. Jackson, J. Tolujevs and Z. Kegenbekov, "Review of Inventory Control Models: A Classification Based on Methods of Obtaining Optimal Control Parameters". Transport and Telecommunication, 21(3), pp. 191-202, 2020.
[42] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, "Openai gym". arXiv preprint arXiv:1606.01540, 2016.
[43] A. Schuderer, S. Bromuri and M. van Eekelen, "Sim-Env: Decoupling OpenAI Gym Environments from Simulation Models". arXiv preprint arXiv:2102.09824, 2021.